Lecture 3 : 행렬/선형대수학의 응용

Fastcampus Math Camp

신승우

Saturday 9th June, 2018

Outline

지난시간에는

- 수식의 파싱
- 벡터
  - 벡터의 정의와 연산
  - 벡터의 선형독립 및 span
  - 좌표계 및 기저
  - 몇 가지 예시

이번시간에는

- 행렬
  - 행렬의 정의와 연산
  - 행렬과 선형변환
- 추상 벡터 공간

Outline

행렬의 정의

행렬

행렬은 벡터의 모임 $M = [\vec{v_i}]$로 정의한다. 이 때, 각 $\vec{v_i}$를 column 벡터라고 한다. $M_{ij} = v_j$의 i번째 원소로 생각한다. 벡터가 n-벡터이고, m개의 column 벡터로 이루어져 있는 행렬을 $m \times n$ 행렬이라고 한다.

이와 비슷하게, row 벡터로 정의할 수도 있다.

행렬의 연산 : 덧셈/뺄셈/스칼라배

- 덧셈/뺄셈
- 스칼라배

행렬의 연산 : 곱셈

- 행렬-행렬 곱셉 : $C = A \times B$ 라 하자. 이 때, A가 $m \times n$ 행렬이고 B가 $p \times k$ 행렬일 때, $n = p$ 여야 곱셉이 가능하며, 결과는 $m \times k$ 행렬이 된다. $C_{ij} = \sum^{m} A_{ik} B_{kj}$ 로 정의된다. 행렬의 곱셈은 일반적으로 교환법칙이 성립하지 않으나, 1) 결합법칙은 성립한다.
- 행렬-벡터 곱셈 : 행렬-행렬 곱셈의 특이한 케이스이다.

위 행렬의 곱셈의 정의에서, $I_{ij} = \delta_{ij}$인 행렬은 임의의 행렬 A에 대해서 2) $AI = IA = A$이다. 이 I를 단위행렬이라고 한다.

Einstein Notation의 소개

앞으로 다룰 대부분의 합은 대개의 경우 벡터나 행렬의 원소들의 곱으로 나타내어진다. 따라서, 공통된 첨자가 있을 경우 합 표기를 따로 명시하지 않는 한 생략한다. 예를 들어서, 두 벡터
$\vec{v} = (v_1, v_2, ..., v_n), \vec{u} = (u_1, u_2, ..., u_n)$의 내적의 경우 원래대로 표기하면

$$\vec{v} \bullet \vec{u} = \sum_{i=1}^{n} v_i u_i \tag{1}$$

이나, 이제부터는

$$\vec{v} \bullet \vec{u} = v_i u_i \tag{2}$$

로 표기할 것이다.

Proof of 1)

m by n 행렬 A, n by p 행렬 B, p by q 행렬 C에 대해서

$$
\begin{aligned}
[(AB)C]_{ij} &= (AB)_{ik}c_{kj} && (3)\\
&= a_{il}b_{lk}c_{kj} && (4)\\
&= a_{ik}(BC)_{kj} = [A(BC)]_{ij} && (5)
\end{aligned}
$$

이다. 따라서 결합법칙이 성립한다.

Proof of 2)

n by n 행렬 A와 I에 대해서,

$$
\begin{aligned}
(AI)_{ij} &= a_{ik} I_{kj} && (6)\\
&= a_{ik} \delta_{kj} && (7)\\
&= a_{ij} = A_{ij} && (8)
\end{aligned}
$$

이며, 비슷하게

$$
\begin{aligned}
(IA)_{ij} &= \delta_{ik} a_{kj} && (9)\\
&= a_{ik} \delta_{kj} && (10)\\
&= a_{ij} = A_{ij} && (11)
\end{aligned}
$$

행렬의 곱셈

- Naive Approach : $O(n^3)$
- Strassen's Algorithm : $O(n^{2.8})$

행렬의 곱셈 : Strassen's Algorithm

$$\mathrm{A} = \begin{bmatrix} A_{11} & A_{12} \\ A_{21} & A_{22} \end{bmatrix}$$

$$\mathrm{B} = \begin{bmatrix} B_{11} & B_{12} \\ B_{21} & B_{22} \end{bmatrix}$$

$$\mathrm{C} = \mathrm{AB} = \begin{bmatrix} C_{11} & C_{12} \\ C_{21} & C_{22} \end{bmatrix}$$

행렬의 곱셈 : Strassen's Algorithm

1) 이 때, naive한 C의 계산은 다음과 같다.

$$C_{11} = A_{11}B_{11} + A_{12}B_{21} \tag{12}$$
$$C_{12} = A_{11}B_{12} + A_{12}B_{22} \tag{13}$$
$$C_{21} = A_{21}B_{11} + A_{22}B_{21} \tag{14}$$
$$C_{22} = A_{21}B_{12} + A_{22}B_{22} \tag{15}$$

이 경우, C를 계산하기 위해서는 8번의 곱하기와 4번의 더하기가 필요하다. 이 때, 곱하기가 비싼 계산이므로, 곱하기의 수를 줄이는 방법에 대해서 생각해 보자.

행렬의 곱셈 : Strassen's Algorithm

$$
\begin{aligned}
M_1 &= (A_{11} + A_{22})(B_{11} + B_{22}) && (16)\\
M_2 &= (A_{21} + A_{22})B_{11} && (17)\\
M_3 &= A_{11}(B_{12} - B_{22}) && (18)\\
M_4 &= A_{22}(B_{21} - B_{11}) && (19)\\
M_5 &= (A_{11} + A_{12})B_{22} && (20)\\
M_6 &= (A_{21} - A_{11})(B_{11} + B_{12}) && (21)\\
M_7 &= (A_{12} - A_{22})(B_{21} + B_{22}) && (22)
\end{aligned}
$$

으로 보조 행렬들을 만들 때, 다음이 성립한다.

행렬의 곱셈 : Strassen's Algorithm

$$C_{11} = M_1 + M_4 - M_5 + M_7 \quad (23)$$
$$C_{12} = M_3 + M_5 \quad (24)$$
$$C_{21} = M_2 + M_4 \quad (25)$$
$$C_{22} = M_1 - M_2 + M_3 + M_6 \quad (26)$$

이를 이용해서, 곱셈을 덜 하고 덧셈을 더 해서 행렬의 곱셈을 최적화할 수 있다. 정확하게는 7번의 곱셈과 18번의 덧셈이 필요하다. 만약 행렬의 크기가 짝수가 아니라면, 필요한 만큼 0을 덧붙여 계산한다.

Complexity of Strassen's Algorithm

$T(n) = 7 * T(n/2) + 18n^2$ 으로 계산된다. 이는 $n = 2^k$로 치환하고 계산하면 $O(n^{log_2 7})$을 얻을 수 있다.

행렬의 연산 : elementary operations

행렬의 elementary operation에는 다음의 3가지가 있다.

- 행/열 바꾸기
- 행/열 더하기
- 특정 행/열에 상수 곱하기

위 연산들을 종합하면, 행렬의 row vector들의 선형결합으로 특정 row vector을 대체하는 것을 말한다. 이는 원 행렬에 적절한 행렬을 곱하여 얻을 수 있다.

행렬의 연산 : elementary operation

row vector $\vec{v_i}$로 이루어진 행렬 A가 있다고 하자. 이 때, row vector $\vec{u_j} = c_{ji}\vec{v_i}$를 row vector로 가지는 행렬 B는
$B = CA$
를 만족한다. 이 때 1) $C_{ij} = c_{ij}$이며, C는 정사각행렬이다.

Proof of 1)

곱셈의 정의를 그대로 쓰면, $(CA)_i = c_{ik} a_k$ 이므로 성립한다.

행렬의 연산 : elementary operation의 예시

- 행/열 바꾸기
$$\begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix} \times \begin{bmatrix} 2 & 1 & 3 \\ -1 & 3 & 2 \end{bmatrix} = \begin{bmatrix} -1 & 3 & 2 \\ 2 & 1 & 3 \end{bmatrix}$$
- 행/열 더하기
$$\begin{bmatrix} 1 & 1 \\ 0 & 1 \end{bmatrix} \times \begin{bmatrix} 2 & 1 & 3 \\ -1 & 3 & 2 \end{bmatrix} = \begin{bmatrix} 1 & 4 & 5 \\ -1 & 3 & 2 \end{bmatrix}$$
- 상수 곱하기
$$\begin{bmatrix} 1 & 0 \\ 0 & 3 \end{bmatrix} \times \begin{bmatrix} 2 & 1 & 3 \\ -1 & 3 & 2 \end{bmatrix} = \begin{bmatrix} 2 & 1 & 3 \\ -3 & 9 & 6 \end{bmatrix}$$
- 선형결합 $\vec{u}_1 = \vec{v}_1, \vec{u}_2 = \vec{v}_1 + 3\vec{v}_2$ 의 경우,
$$\begin{bmatrix} 1 & 0 \\ 1 & 1 \end{bmatrix} \times \begin{bmatrix} 1 & 0 \\ 0 & 3 \end{bmatrix} \times \begin{bmatrix} 2 & 1 & 3 \\ -1 & 3 & 2 \end{bmatrix} = \begin{bmatrix} 2 & 1 & 3 \\ -1 & 10 & 9 \end{bmatrix}$$

행렬의 연산 : 가우스 소거법(실습)

가우스 소거법은 elementary operation의 연장선상으로, 행렬을 reduced row echelon 행렬로 만드는 것을 말한다. reduced row echelon 행렬을 정의하기 위해서 우선 행렬의 i번째 행의 선행 계수를 다음과 같이 정의하자.

$j(i) = min(j \leq n, M_{ij} \neq 0)$

이 때, 다음 조건을 만족시키는 행렬을 row echelon 행렬이라고 하며, 각 행의 선행 계수에 해당하는 원소가 1이며 그 열에서는 1 하나를 빼고 모든 원소가 0인 경우 reduced row echelon 행렬이라고 한다.

- 행의 모든 원소가 0이면, 그 행보다 아래에 있는 행의 모든 원소 또한 0이다.
- $i < k, j(i) < j(k)$ 이다.

즉, 0이 아닌 숫자가 사다리꼴로 배열되어 있는 행렬을 말한다.

행렬의 연산 : 가우스 소거법(실습)

가우스 소거법은 다음과 같다.

- j번째 행에 대해서, 첫번째 행부터 순차적으로
  - 선행 계수가 존재하는 가장 작은 열수를 찾는다.
  - 현재 행과 찾아진 행을 치환한다.
  - 선행계수에 해당하는 열에서 현재 행보다 아래 행들의 해당 열을 0으로 만든다.

가우스 소거법 실습 I

지금부터 30분 동안 진행합니다. 짜는 과정에서 다음의 보조함수들을 짜는 것을 추천합니다.

- 선행 계수가 존재하는 가장 작은 열을 찾는 함수(_find_leading_row)
- elementary operation들과 그에 해당하는 행렬
  - 행 바꾸기 (change_row)
  - 행에 상수 곱하기 (multiply_row)
  - 행에 다른 행 더하기 (add_row)
- Identity 행렬 출력하는 static method(PyMatrix.identity)

출력은 리스트와 PyMatrix 인스턴스 하나로, 다음과 같습니다.

- 리스트 : 어떤 elementary operation을 했는지 대응되는 행렬의 리스트
- 가우스 소거의 결과 행렬

다 한 후, 시간이 남으면 나온 결과물을 검증하는 코드도 짜서 검증해보시면 좋습니다.

행렬의 연산 : trace, transpose

- trace : $tr(A) = \sum_i A_{ii}$
- transpose : $(A^T)_{ij} = A_{ji}$

임의의 행렬 A,B에 대해서

- $(AB)^T = B^T A^T$
- $tr(AB) = tr(BA)$

Proof of 1)

$$
\begin{aligned}
((AB)^T)_{ij} &= (AB)_{ji} && (27)\\
&= a_{jk} b_{ki} && (28)\\
&= a^T_{kj} b^T_{ik} && (29)\\
&= b^T_{ik} a^T_{kj} && (30)\\
&= (B^T A^T)_{ij} && (31)
\end{aligned}
$$

이므로 성립한다.

Proof of 2)

$$
\begin{aligned}
tr(AB) &= \sum AB_{ii} && (32)\\
&= a_{ik}b_{ki} && (33)\\
&= b_{ki}a_{ik} && (34)\\
&= \sum BA_{kk} = tr(BA) && (35)
\end{aligned}
$$

행렬의 연산 : determinant

행렬식은 재귀적으로 정의되며, 정사각행렬에서만 정의된다. 행렬 A의 행렬식 $det(A)$는

- 행렬의 size가 (1,1) : A_{11}
- 행렬의 size가 (n,n) : $\sum_{i=1}^{n}(-1)^n det(minor(A,1,i))$ 여기서, $minor(A,i,j)$는 i번째 row와 j번째 column을 제거한 행렬을 말한다.

1) 두 행렬 A,B에 대해서 $det(AB) = det(A)det(B)$ 이다.

행렬의 연산 : 역행렬

행렬 A의 역행렬은 A^{-1} 이라고 표기하며, $AA^{-1} = A^{-1}A = I$ 를 만족한다.
A가 역행렬을 가지면, A는 invertible matrix라고 한다.
역행렬은 다음의 성질들을 가진다.

- A^{-1}이 존재하려면 A는 정사각행렬이여야 한다.
- $(A^{-1})^{-1} = A$
- A의 역행렬은 유일하다.
- A의 역행렬이 있다면 $(A^T)^{-1} = (A^{-1})^T$
- 역행렬이 있는 행렬은 행렬식이 0이 아니며, 역도 성립한다.

Proof of 3)

서로 다른 두 행렬 B,C가 A의 역행렬이라고 하면,
$B = BI = B(AC) = (BA)C = C$
이므로 B=C이다. 따라서 모순이므로 역행렬은 유일하다.

Proof of 4)

행렬 A에 대해서,
$(AA^{-1})^T = (A^{-1})^T A^T = I^T = I$ 이므로 성립한다.

역행렬을 찾는 법

행렬 A에 대해서 역행렬을 찾는 법을 생각해보자. 먼저, A가 $\vec{v_i}$의 row vector로 이루어져 있다고 생각하자. 이 때, 어떤 elementary operation들을 수행하여 A를 I로 만들 수 있다면, 그 elemantary operation들에 해당하는 행렬이 역행렬이 될 것이다.
예를 들어서,

$$\begin{bmatrix} 1 & 1 \\ 0 & 1 \end{bmatrix} \times \begin{bmatrix} 1 & -1 \\ 0 & 1 \end{bmatrix} = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}$$

이므로

$$\begin{bmatrix} 1 & 1 \\ 0 & 1 \end{bmatrix}^{-1} = \begin{bmatrix} 1 & -1 \\ 0 & 1 \end{bmatrix}$$

이다. 즉, 역행렬을 찾는 것은 elementary operation들을 수행해서 행렬을 I로 만드는 것과 같다. 이제 본격적으로 어떻게 역행렬을 찾는지 알아보자.

역행렬 찾기

행렬 A에 대해서 다음의 과정을 시행한다.

- for i $= 1, 2, \ldots, n$,
  - $a_{ii} = 1$이 되게 한다. 이 때,
    - 1) 만약 a_{ji}이 모두 0이라면, elementary operation을 이용하여 $a_{ii} = 1$이 되게 하는 것은 불가능하다. 따라서 역행렬도 존재하지 않는다.
    - 하나라도 0이 아니라면, 그 행을 이용하여 $a_{ii} = 1$로 만든다.
  - 그 후 i가 아닌 모든 j에 대해서, a_{ji}가 0이 되도록 만든다.

선형변환

$V = \mathbb{R}^n$에서 $W = \mathbb{R}^m$으로의 선형변환은 V의 원소를 벡터공간 W의 원소로 대응시키는 함수 중 다음을 만족하는 함수를 말한다.

- $f(\vec{u} + \vec{v}) = f(\vec{u}) + f(\vec{v})$
- $f(c\vec{u}) = cf(\vec{u})$

선형변환과 행렬

선형변환은 행렬로 볼 수 있다. 더 정확하게는, 1) $V = \mathbb{R}^n$에서 $W = \mathbb{R}^m$으로의 선형변환은 m by n 행렬로 볼 수 있다.

Construction from Linear Transformation to Matrix I

V의 기저 $\{\vec{v}_i\}$를 생각하자. 이 때, V의 임의의 원소인 $\vec{v}$는 다음과 같이 쓸 수 있다.

$$\vec{v} = c_i \vec{v}_i \tag{36}$$

이 때, 선형변환 $f : V \to W$에 대해서 $f(\vec{v})$는

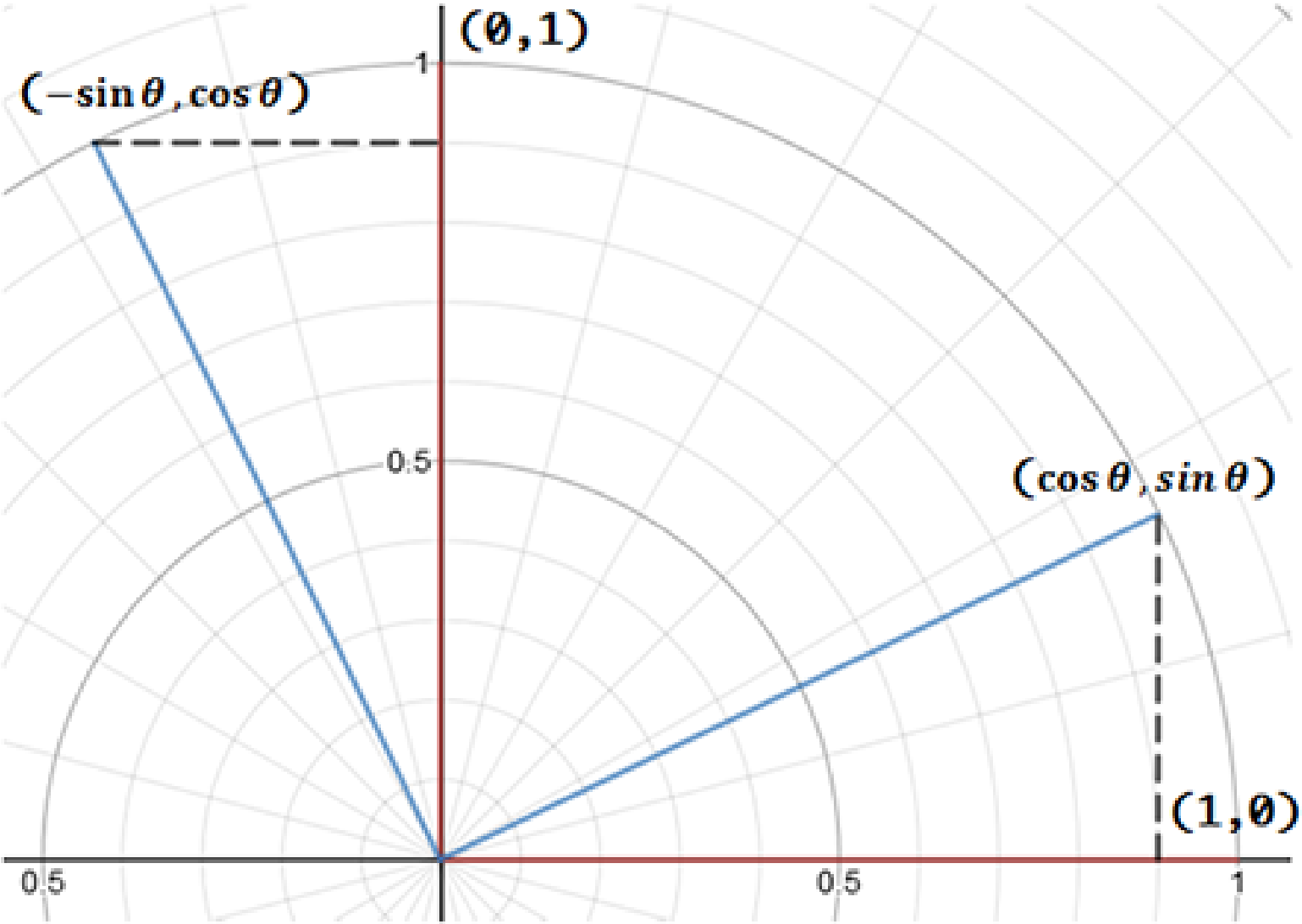
$$f(\vec{v}) = f(c_i \vec{v}_i) = c_i f(\vec{v}_i) \tag{37}$$

로 생각할 수 있다. 이제, W의 기저 $\{\vec{w}_i\}$를 생각하자. $f(\vec{v}_i)$는 W의 원소이므로 다음과 같이 쓸 수 있다.

$$f(\vec{v}_i) = d_i j \vec{w}_j \tag{38}$$

이에 착안해서, d_{ij}를 이용해서 행렬 D를 만들면, n by m 행렬이 된다. 이 행렬을 Transformation Matrix라고 한다. 이 행렬을 이용하면, $\vec{v} \in V$에 대해서 $f(\vec{v}) = D\vec{v}$ 임을 알 수 있다.

선형변환의 예시 : 2차원 회전변환



선형변환의 예시 : 2차원 회전변환

$$T_\theta = \begin{bmatrix} cos\theta & -sin\theta \\ sin\theta & cos\theta \end{bmatrix}$$

행렬의 연산을 이용한 선형변환

선형변환이라는 함수가 행렬으로 표현가능하므로, 행렬의 연산을 이용해서 선형변환을 원하는 대로 만들 수 있다. 여기서는 크게 두 가지를 살펴보고자 한다.

- 합성함수와 행렬의 곱
- 역함수와 역행렬

이 두 가지 예시를 회전변환을 이용해서 살펴보고자 한다.

행렬 곱과 함성함수

두 회전변환

$$T_\alpha = \begin{bmatrix} cos\alpha & -sin\alpha \\ sin\alpha & cos\alpha \end{bmatrix}$$

$$T_\beta = \begin{bmatrix} cos\beta & -sin\beta \\ sin\beta & cos\beta \end{bmatrix}$$

가 있을 때, 두 변환행렬의 곱은

$$T_\alpha T_\beta = \begin{bmatrix} cos\alpha & -sin\alpha \\ sin\alpha & cos\alpha \end{bmatrix} \times \begin{bmatrix} cos\beta & -sin\beta \\ sin\beta & cos\beta \end{bmatrix} =$$

$$\begin{bmatrix} cos\alpha cos\beta - sin\alpha sin\beta & -cos\alpha sin\beta - sin\alpha cos\beta \\ cos\alpha sin\beta + sin\alpha cos\beta & cos\alpha cos\beta - sin\alpha sin\beta \end{bmatrix}$$

인데, 이는 삼각함수의 합차공식을 이용하면 $T_{\alpha+\beta}$임을 알 수 있다.

역행렬과 역함수

T_α의 역행렬은 다음과 같이 구할 수 있다.

$T_\alpha T^{-1} = \begin{bmatrix} cos\alpha & -sin\alpha \\ sin\alpha & cos\alpha \end{bmatrix} \times \begin{bmatrix} cos\alpha & sin\alpha \\ -sin\alpha & cos\alpha \end{bmatrix} = I_2$ 인데, 이에서

$T^{-1} = T_{-\alpha}$임을 알 수 있다.

Definition of Eigen*

행렬 A에 대해서, 다음을 만족하는 λ와 $\vec{x}$들을 각각 고유값(eigenvalue)과 고유벡터(eigenvector)라고 한다.
$A\vec{x} = \lambda\vec{x}$
위 식을 다시 쓰면
$(A - \lambda I)\vec{x} = 0$
이며, 여기서 $det(A - \lambda I) = 0$를 특성방정식이라 한다. 이 때 특성방정식의 해는 고유값이 되며, 이에 따라 고유벡터를 계산할 수 있다.
고유벡터들로 span되는 공간을 eigenspace라 한다.

Motivation of Eigen* I

위에서 다룬 회전변환 행렬 T를 생각해 보자.

$$T_\theta = \begin{bmatrix} cos\theta & -sin\theta \\ sin\theta & cos\theta \end{bmatrix}$$

이 때, 회전의 축을 어떻게 구할 수 있을까? 회전의 축은 회전 전후에도 변화가 없을 것이다. 따라서, 회전의 축을 벡터 $\vec{a}$라 하면, $T\vec{a} = \vec{a}$ 임이 성립해야 한다. 이 경우, 이를 만족하는 벡터 a는 0벡터 뿐이다. 그렇다면 이는 무슨 의미를 가질까? 이는 회전축이 z축이므로, 이를 나타내는 것으로 생각할 수 있다. 이는 다음의 3차원 회전변환 행렬을 생각해보면 조금 더 명확해진다.

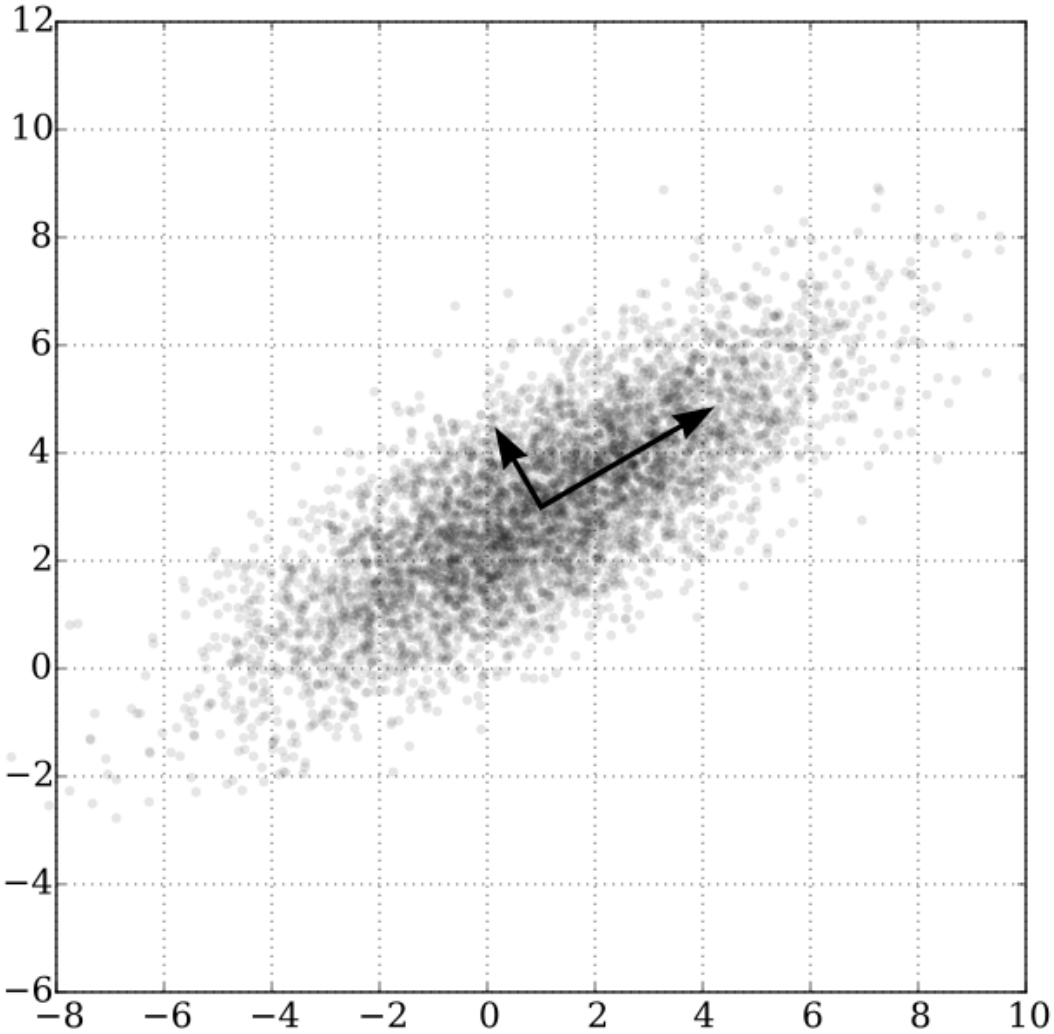
$$T_\theta = \begin{bmatrix} cos\theta & -sin\theta & 0 \\ sin\theta & cos\theta & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

이 경우 위와 같은 방정식을 풀면 z는 어떠한 수여도 가능하고, x와 y는 0임을 알 수 있다. 즉, z축이 회전의 축임을 알 수 있다.

Motivation of Eigen* II

이는 일반적인 선형변환에서도 같다. 어떤 선형변환 R에 대해서, $R\vec{a} = \vec{a}$가 성립한다면, 그 벡터는 선형변환에 대해서 불변이다. 즉, 어떠한 행렬의 eigenvector는 그 행렬에 대응하는 선형변환의 축이라고 볼 수 있다.

직관적 이해 : Principal Component Analysis



Properties of Eigen*

일반적으로, 다음 성질들이 성립한다.

- A^T의 고유값과 고유벡터는 A와 같다.
- $A^T A$의 고유벡터는 A와 같고, 고유값은 제곱값이다.
- det(A)는 모든 고유값의 곱이다.

Proof of 3) I

어떤 행렬 A의 고유벡터를 $\vec{v}_i$, 고유값을 λ_i라 하자. 그러면 다음이 성립한다.

$$
\begin{aligned}
A\begin{bmatrix}\vec{v}_1 & \vec{v}_2 & \dots & \vec{v}_n\end{bmatrix} &= A\begin{bmatrix}A\vec{v}_1 & A\vec{v}_2 & \dots & A\vec{v}_n\end{bmatrix} && (39)\\
&= A\begin{bmatrix}\lambda_1\vec{v}_1 & \lambda_2\vec{v}_2 & \dots & \lambda_n\vec{v}_n\end{bmatrix} && (40)\\
&= \begin{bmatrix}\lambda_1\vec{v}_1 & \lambda_2\vec{v}_2 & \dots & \lambda_n\vec{v}_n\end{bmatrix}\begin{bmatrix}\lambda_1 & 0 & \dots & 0\\ 0 & \lambda_2 & \dots & 0\\ 0 & 0 & \dots & \lambda_n\end{bmatrix} && (41)
\end{aligned}
$$

이다. 이 때, det(AB) = det(A) det(B)임을 이용하면

Proof of 3) II

$$det(A)det\left(\begin{bmatrix}\vec{v}_1 & \vec{v}_2 & \dots & \vec{v}_n\end{bmatrix}\right) \tag{42}$$

$$= det\left(\begin{bmatrix}\lambda_1\vec{v}_1 & \lambda_2\vec{v}_2 & \dots & \lambda_n\vec{v}_n\end{bmatrix}\right) det\left(\begin{bmatrix}\lambda_1 & 0 & \dots & 0\\ 0 & \lambda_2 & \dots & 0\\ 0 & 0 & \dots & \lambda_n\end{bmatrix}\right) \tag{43}$$

$$\tag{44}$$

이므로, det(A)는 모든 고유값의 곱이 된다.

Eigendecomposition

Eigendecomposition은 어떤 행렬의 eigenvector들이 서로 선형독립일 때 가능하다. 위 증명에서,

$$A\begin{bmatrix}\vec{v}_1 & \vec{v}_2 & \dots & \vec{v}_n\end{bmatrix} = \begin{bmatrix}\lambda_1\vec{v}_1 & \lambda_2\vec{v}_2 & \dots & \lambda_n\vec{v}_n\end{bmatrix}\begin{bmatrix}\lambda_1 & 0 & \dots & 0\\ 0 & \lambda_2 & \dots & 0\\ 0 & 0 & \dots & \lambda_n\end{bmatrix} \quad (45)$$

에서, 좌변의 $\begin{bmatrix}\vec{v}_1 & \vec{v}_2 & \dots & \vec{v}_n\end{bmatrix}$ 가 만약 역행렬을 가진다면, 다음과 같이 A를 분해할 수 있다.

$A = QLQ^{-1}$

여기서 Q는 $\begin{bmatrix}\vec{v}_1 & \vec{v}_2 & \dots & \vec{v}_n\end{bmatrix}$ 이고, L은 $\begin{bmatrix}\lambda_1 & 0 & \dots & 0\\ 0 & \lambda_2 & \dots & 0\\ 0 & 0 & \dots & \lambda_n\end{bmatrix}$ 이다.

Outline

일차방정식 Solver 만들기(실습)

일차방정식들 $a_ijx_j = b_i, i, j = 1, 2, ..., n$을 다음과 같이 쓸 수 있다.
$A\vec{x} = \vec{b}$
여기서 $A_{ij} = a_{ij}, \vec{x}[i] = x_i, \vec{b}[i] = b_i$이다. 이때, 일차방정식의 해 $\vec{x}$는 다음과 같은 과정을 통해서 구할 수 있다.

- A의 역행렬을 구한다.
  - 만약 실패하면 해가 없는 것이다.
  - 성공하면, $A^{-1}\vec{b}$ 를 계산한다.

Outline

Generalized Arithematic Operations

이때까지 벡터, 행렬 등의 *선형대수학적* 객제들의 더하기와 상수배의 operation에 대해서 다루었다. 이제 이 연산을 확장해 보자. 즉, 이제부터는 더하기, 곱하기 등은 굳이 사칙연산일 필요가 없다. 예컨대, 함수끼리의 곱셈을 다음과 같이 정의할 수도 있다.

$\int f \bullet g dx$

더하기 역시 정의하기 나름이다. 이제부터 이러한 일반화된 사칙연산을 고려하기 위한 체계를 세워보겠다.

항등원과 역원

어떤 집합 S에서의 이항연산 f에 대해서,

- 집합 S의 모든 원소 s에 대해서 f(s,i) = f(i,s) = s 이면 i를 그 연산의 항등원이라고 한다.
- 집합 S의 어떤 원소 s에 대해서 f(s, t) = f(t, s) = i 이면 t를 s의 역원이라고 한다.

Generalized Vector Space

어떤 집합 F 위에서 정의된 벡터공간 V는 어떤 집합 V와 F의 원소 a,b 와 V의 원소 $\vec{v}, \vec{u}$에 대해서 다음이 성립하는 벡터연산 더하기와 스칼라곱, 그리고 덧셈의 역원으로 정의된다. 이 때, F를 이 벡터공간의 스칼라라고 한다.

- 벡터덧셈의 교환법칙 / 결합법칙
- 벡터덧셈의 항등원 / 역원
- 스칼라의 곱셈에서의 항등원 1에 대해서, $1\vec{v} = \vec{v}$
- $(ab)\vec{v} = a(b\vec{v})$
- $a(\vec{v} + \vec{u}) = a\vec{v} + a\vec{u}$
- $(a + b)\vec{v} = a\vec{v} + b\vec{v}$

여기서, 스칼라곱과 벡터-스칼라곱이나 스칼라끼리의 합과 벡터-벡터간의 합은 다른 operation이다.

Examples of an Abstract Vector Space

- Polynomials with degree $\leq n$
- Matrices
- Linear Transformations
- Functions from a specific domain (will be revisited after few weeks)
- Random Variables (will be revisited after few weeks)

예시 : $\mathbb{R}^2$에서 $\mathbb{R}^2$로의 선형변환

$\mathbb{R}^2$에서 $\mathbb{R}^2$로의 선형변환은 2 by 2 행렬로 나타내어질 수 있다. 따라서, 행렬의 덧셈과 실수배를 이용하여 선형변환의 벡터공간을 정의할 수 있다. 이제부터 이 공간의 기저와 좌표, 그리고 2차원 좌표계에서 선형변환 벡터들이 어떠한 의미를 가지는지를 알아볼 것이다.

예시 : $\mathbb{R}^2$에서 $\mathbb{R}^2$로의 선형변환

먼저, 다음의 행렬들을 생각해 보자.

$\vec{e_1} = \begin{bmatrix} 1 & 0 \\ 0 & 0 \end{bmatrix}, \vec{e_2} = \begin{bmatrix} 0 & 1 \\ 0 & 0 \end{bmatrix}, \vec{e_3} = \begin{bmatrix} 0 & 0 \\ 1 & 0 \end{bmatrix}, \vec{e_3} = \begin{bmatrix} 0 & 0 \\ 0 & 1 \end{bmatrix}$

이 행렬들은 명백하게 2 by 2 행렬이므로, $\mathbb{R}^2$에서 $\mathbb{R}^2$로의 선형변환이다.

이들 벡터는 선형독립일까? 또, 위 4개의 벡터는 기저가 될까?

Linear Independence of an Abstract Vector Space

위 4개의 벡터가 선형독립이기 위해서는 $c_i e_i$을 만족하는 c_i가 0뿐이여야 한다. 위 경우,

$c_i \vec{e_i} = \begin{bmatrix} c_1 & c_2 \\ c_3 & c_4 \end{bmatrix}$ 이므로 선형독립임을 알 수 있다.

또한, 임의의 2 by 2 행렬을 c_i를 적절히 조정하여 만들 수 있으므로, 기저임을 알 수 있다.

예시 : $\mathbb{R}^2$에서 $\mathbb{R}^2$로의 선형변환

위와 같이 선형변환의 기저를 찾은 것은 어떤 의미가 있을까?
이는 이제부터 우리가 $\mathbb{R}^2$의 한 점에서 $\mathbb{R}^2$의 한 점으로 대응시키는 선형변환을 언제나 4개의 선형변환을 조합하여 하는 것으로 이해할 수 있다는 점이다.

Topics for next lecture

- Parser Revisited : extending parser
- 함수의 극한과 미분의 정의. 여러 함수의 미분 및 symbolic 미분기 구현.